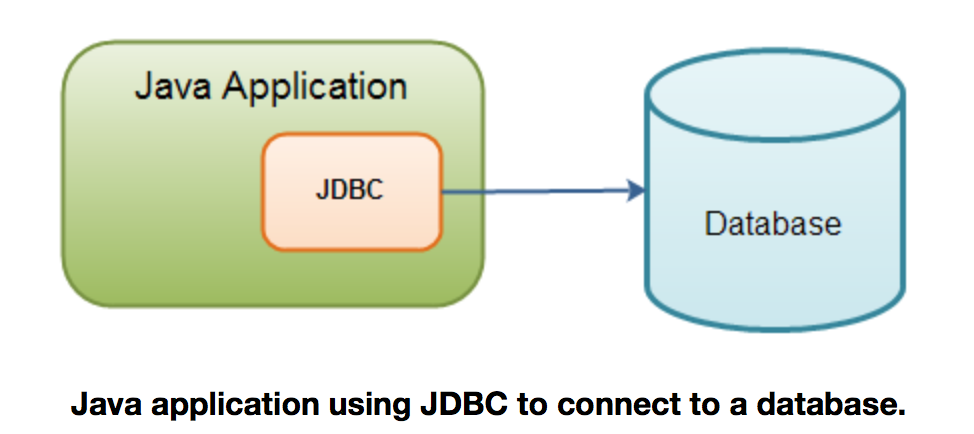
**JDBC**

[**http://tutorials.jenkov.com/jdbc/index.html**](http://tutorials.jenkov.com/jdbc/index.html)

**Java JDBC:**

The Java JDBC API enables Java applications to connect to relational databases via a standard API, so your Java applications become independent of the databases the application uses.



JDBC standardize how to connect to database, how to execute queries against it, how to navigate the result of such query, and how to execute updates in the database.

**JDBC Overview:**

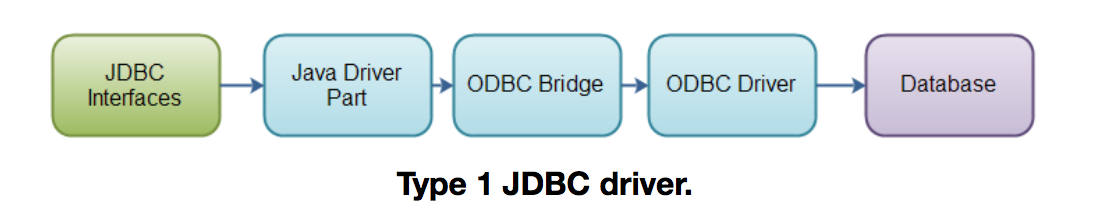
JDBC API consists of following core parts:

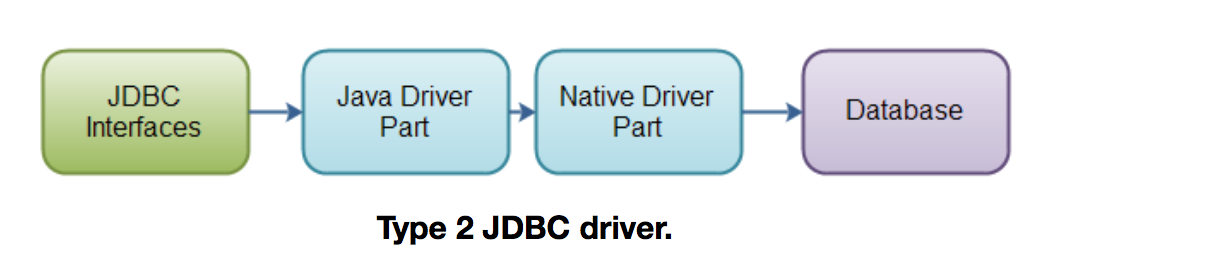
* JDBC Drivers
  + A JDBC driver is a collection of Java classes that enables you to connect to a certain database. For instance, MYSQL will have its own JDBC driver. A JDBC driver implements a lot of the JDBC interfaces. When your code uses a given JDBC driver, it actually just uses the standard JDBC interfaces.
* Connections
  + Once the JDBC driver is loaded and initialized, you need to connect to database. You do so by obtaining a Connection to the database via the JDBC API and the load driver. All communication with database happens via a connection. An application can have more than one connection open to a database at a time. This is actually very common.
* Statements
  + A Statement is what you use to execute queries and updates against the database. There are few different types of statements you can use. Each statement corresponds to a single query or update.
* ResultSets
  + When you perform a query against the database you get back a ResultSet. You can then traverse this ResultSet to read the result of the query.

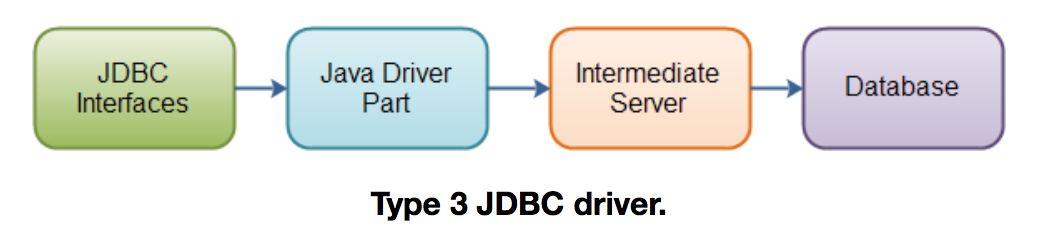
**JDBC Drivers:**

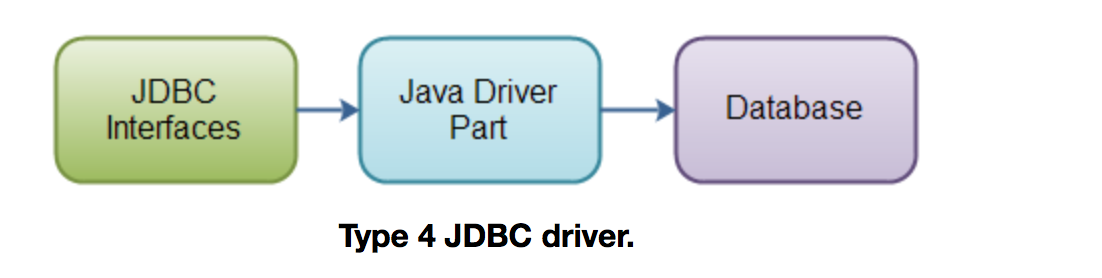
There are 4 types of JDBC drivers:

* Type 1: JDBC-ODBC bridge driver
* Type 2: Java + Native code driver
* Type 3: All Java + Middleware translation driver
* Type 4: All Java driver









**JDBC Database Connections:**

**Loading the JDBC Driver:**

The first thing you need to do before you can open a database connection is to load the JDBC driver for the database. Actually, from Java 6 this is no longer necessary, but doing so will not fail. You load the JDBC driver like this:

|  |
| --- |
| Class.forName("driverClassName"); |

You only have to load the driver once. You do not need to load it before every connection opened. Only before the first connection opened.

**Opening the Connection:**

To open a database connection you use the java.sql.DriverManager class. You call its getConnection() method, like this:

|  |
| --- |
| String url = "jdbc:h2:~/test"; //database specific url.  String user = "sa";  String password = "";  Connection connection =  DriverManager.getConnection(url, user, password); |

The url is the url to your database. You should check the documentation for your database and JDBC driver to see what the format is for your specific database. The url shown above is for a H2Database.

The user and password parameters are the user name and password for your database.

**Closing the Connection:**

Once you are done using the database connection you should close it. This is done by calling the Connection.close() method, like this:

|  |
| --- |
| connection.close(); |

**JDBC: Query the Database**

Querying a database means searching through its data. You do so be sending SQL statements to the database. To do so, you first need an [open database connection](http://tutorials.jenkov.com/jdbc/connection.html). Once you have an open connection, you need to create a Statement object, like this:

|  |
| --- |
| Statement statement = connection.createStatement(); |

Once you have created the Statement you can use it to execute SQL queries, like this:

|  |
| --- |
| String sql = "select \* from people";  ResultSet result = statement.executeQuery(sql); |

When you execute an SQL query you get back a ResultSet. The ResultSet contains the result of your SQL query. The result is returned in rows with columns of data. You iterate the rows of the ResultSet like this:

|  |
| --- |
| while(result.next()) {  String name = result.getString("name");  long age = result.getLong ("age");} |

The ResultSet.next() method moves to the next row in the ResultSet, if there are anymore rows. If there are anymore rows, it returns true. If there were no more rows, it will return false.

You need to call next() at least one time before you can read any data. Before the first next() call the ResultSet is positioned before the first row.

You can get column data for the current row by calling some of the getXXX() methods, where XXX is a primitive data type. For instance:

|  |
| --- |
| result.getString ("columnName");  result.getLong ("columnName");  result.getInt ("columnName");  result.getDouble ("columnName");  result.getBigDecimal("columnName"); |

|  |
| --- |
| result.getString (1);  result.getLong (2);  result.getInt (3);  result.getDouble (4);  result.getBigDecimal(5);  etc. |

|  |
| --- |
| result.close();  statement.close(); |

**JDBC: Update the Database**

1. Update record values
2. Delete records

|  |
| --- |
| Statement statement = connection.createStatement();  String sql = "update people set name='John' where id=123";  int rowsAffected = statement.executeUpdate(sql); |

|  |
| --- |
| Statement statement = connection.createStatement();  String sql = "delete from people where id=123";  int rowsAffected = statement.executeUpdate(sql); |

**JDBC: ResultSet**